ENV 790.30 - Time Series Analysis for Energy Data | Spring 2023

Assignment 8 - Due date 03/27/23

Yuxiang Ren

Directions

You should open the .rmd file corresponding to this assignment on RStudio. The file is available on our class repository on Github. And to do so you will need to fork our repository and link it to your RStudio.

Once you have the project open the first thing you will do is change "Student Name" on line 3 with your name. Then you will start working through the assignment by **creating code and output** that answer each question. Be sure to use this assignment document. Your report should contain the answer to each question and any plots/tables you obtained (when applicable).

When you have completed the assignment, **Knit** the text and code into a single PDF file. Rename the pdf file such that it includes your first and last name (e.g., "LuanaLima_TSA_A08_Sp22.Rmd"). Submit this pdf using Sakai.

Set up

Some packages needed for this assignment: `forecast`,`tseries`,`smooth`. Do not forget to load them before running your script, since they are NOT default packages.

```
#Load/install required package here
#install.packages("tinytex")
library(tinytex)
library(forecast)
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(tseries)
library(smooth)
```

```
## Loading required package: greybox

## Package "greybox", v1.0.7 loaded.

## This is package "smooth", v3.2.0
```

```
library(tidyr)
```

```
##
## Attaching package: 'tidyr'
## The following object is masked from 'package:greybox':
##
##     spread
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:greybox':
##
##     hm

## The following objects are masked from 'package:base':
##
##     date, intersect, setdiff, union
```

```
library(ggplot2)
library(kableExtra)
```

```
## Warning in !is.null(rmarkdown::metadata$output) && rmarkdown::metadata$output
## %in% : 'length(x) = 2 > 1' in coercion to 'logical(1)'

##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

Importing and processing the data set

Consider the data from the file "inflowtimeseries.txt". The data corresponds to the monthly inflow in m^3/s for some hydro power plants in Brazil. You will only use the last column of the data set which represents one hydro plant in the Amazon river basin. The data span the period from January 1931 to August 2011 and is provided by the Brazilian ISO.

For all parts of the assignment prepare the data set such that the model consider only the data from January 2000 up to December 2009. Leave the year 2010 of data (January 2010 to December 2010) for the out-of-sample analysis. Do **NOT** use data fro 2010 and 2011 for model fitting. You will only use it to compute forecast accuracy of your model.

Part I: Preparing the data sets

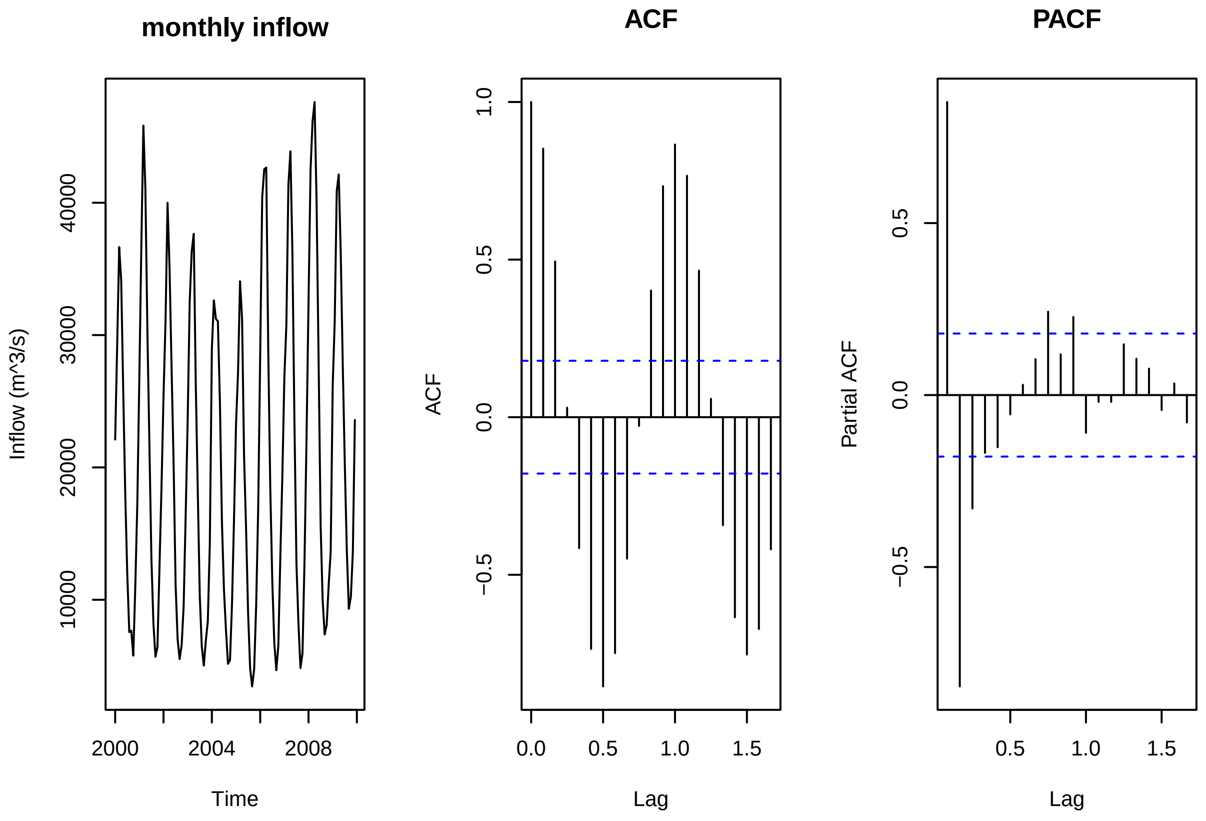
Q1

Read the file into a data frame. Prepare your time series data vector such that observations start in January 2000 and end in December 2009. Make you sure you specify the **start=** and **frequency=** arguments. Plot the time series over time, ACF and PACF.

```
raw <- read.csv(
  file="./Data/inflowtimeseries.txt",
  header=FALSE, sep = "\t")
raw_clean <- raw %>% separate(col = V1,  into = c("month", "year", "inflow"), sep = " ") %>%
  mutate(Date = paste(year, month, "01", sep = "-")) %>% select(Date, V15)
```

```
## Warning: Expected 3 pieces. Additional pieces discarded in 971 rows [1, 2, 3, 4, 5, 6,
## 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, ...].
```

```
#Date period
raw_clean$Date <- ymd(raw_clean$Date)
raw_inflow_09 <- filter(raw_clean, Date >= "2000-01-01" & Date <= "2009-12-01")
raw_inflow_10 <- filter(raw_clean, Date >= "2000-01-01" & Date <= "2010-12-01")
raw_inflow_2010 <- filter(raw_clean, Date >= "2010-01-01" & Date <= "2010-12-01")
#ts
ts_09 <- ts(raw_inflow_09$V15, start = c(2000,1), frequency = 12)
par(mfrow = c(1,3))
plot(ts_09, main = "monthly inflow",ylab ="Inflow (m^3/s)")
acf(ts_09, main = "ACF")
pacf(ts_09, main = "PACF")
```

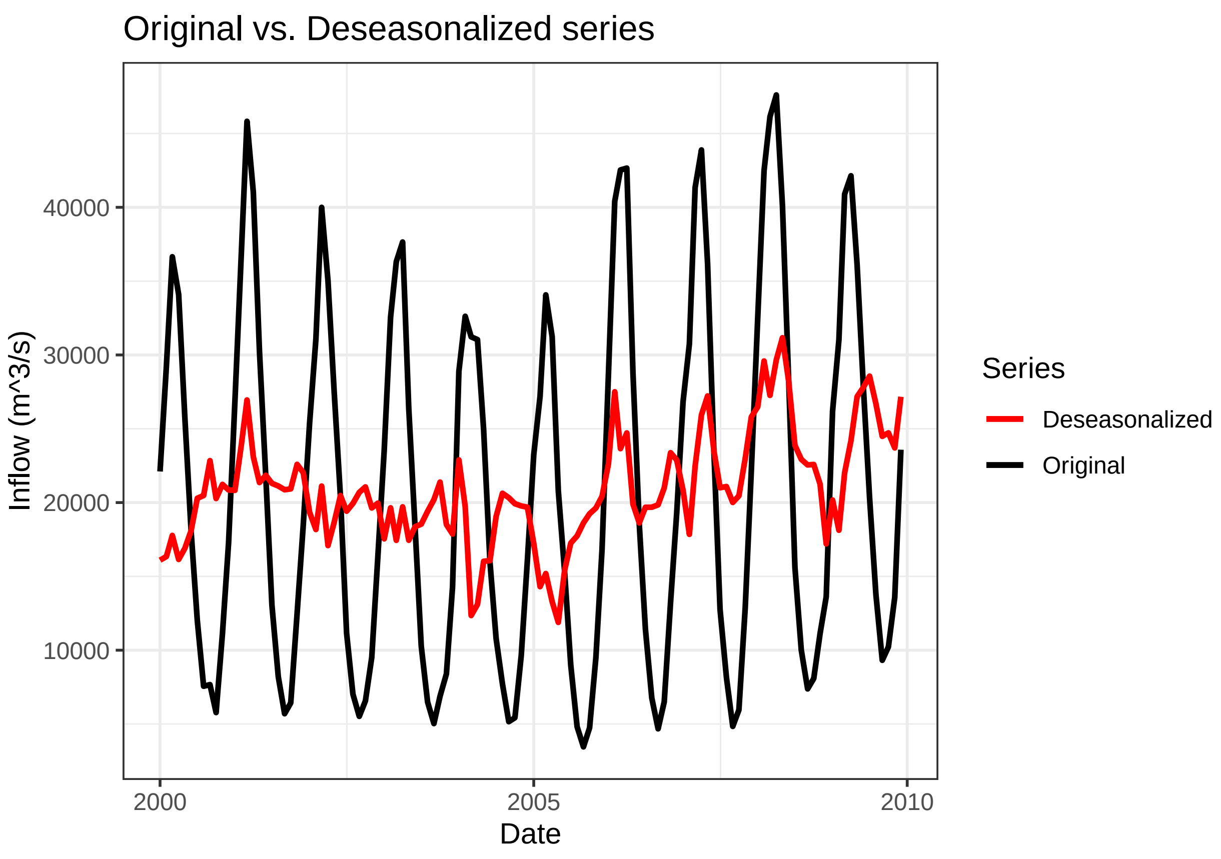


Q2

Using the *decompose*() or *stl*() and the *seasadj*() functions create a series without the seasonal component, i.e., a deseasonalized inflow series. Plot the deseasonalized series and original series together using ggplot, make sure your plot includes a legend. Plot ACF and PACF for the deaseasonalized series. Compare with the plots obtained in Q1.

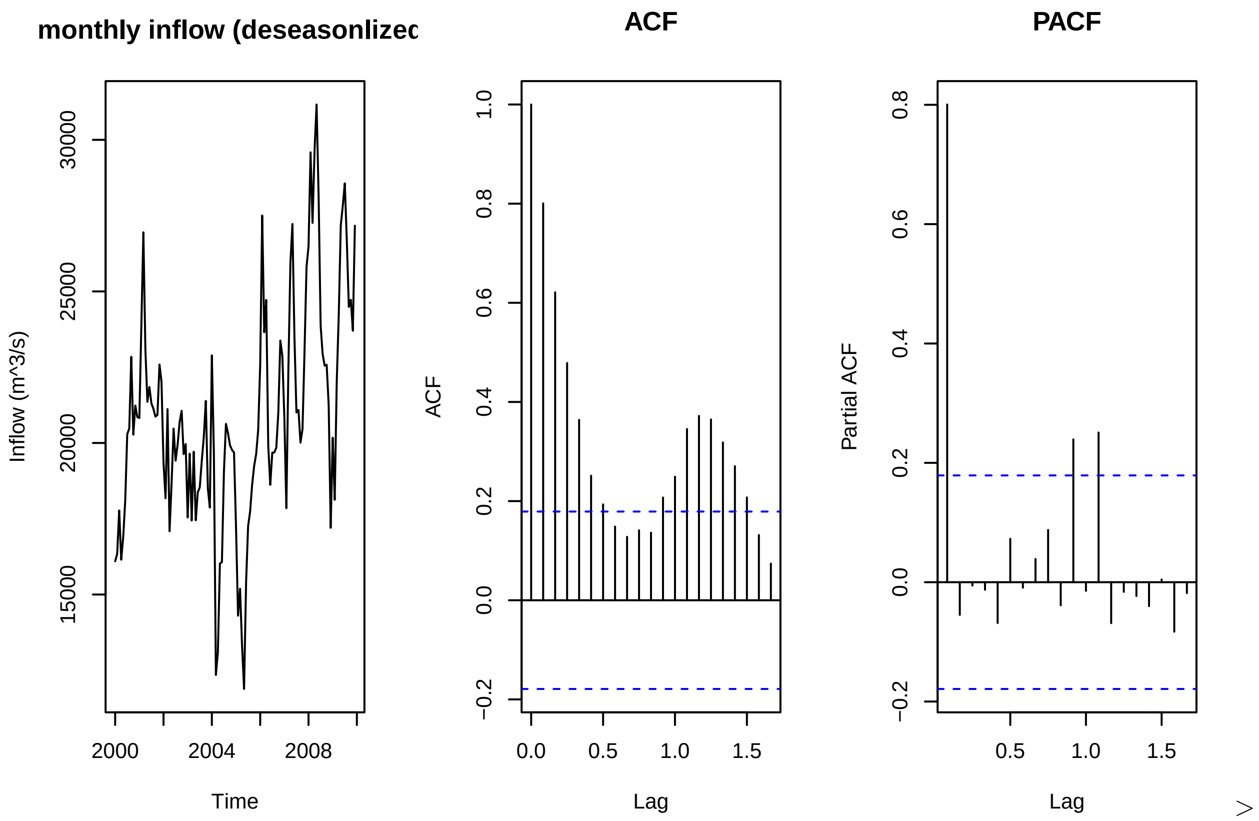
```
inflow09_decomp <- stl(ts_09, s.window = "periodic")
inflow09_desea <- seasadj(inflow09_decomp)
inflow09_df <- data.frame(Date = raw_inflow_09$Date, Original = raw_inflow_09$V15, Deseasonalized = infl
#plot:original vs deseasonalized
ggplot(inflow09_df, aes(x = Date)) +
  geom_line(aes(y = Original, color = "Original"), size = 1) +
  geom_line(aes(y = Deseasonalized, color = "Deseasonalized"), size = 1) +
  scale_color_manual(values = c("Original" = "black", "Deseasonalized" = "red")) +
  labs(x = "Date", y = "Inflow (m^3/s)", color = "Series") +
  ggtitle("Original vs. Deseasonalized series") +
  theme_bw()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
```



```
#ACF,PACF deseasonalized
ts_09_desea <- ts(inflow09_desea, start = c(2000,1), frequency = 12)
par(mfrow = c(1,3))
plot(ts_09_desea, main = "monthly inflow (deseasonlized)",ylab ="Inflow (m^3/s)" )
```

```
acf(ts_09_desea, main = "ACF")
pacf(ts_09_desea, main = "PACF")
```



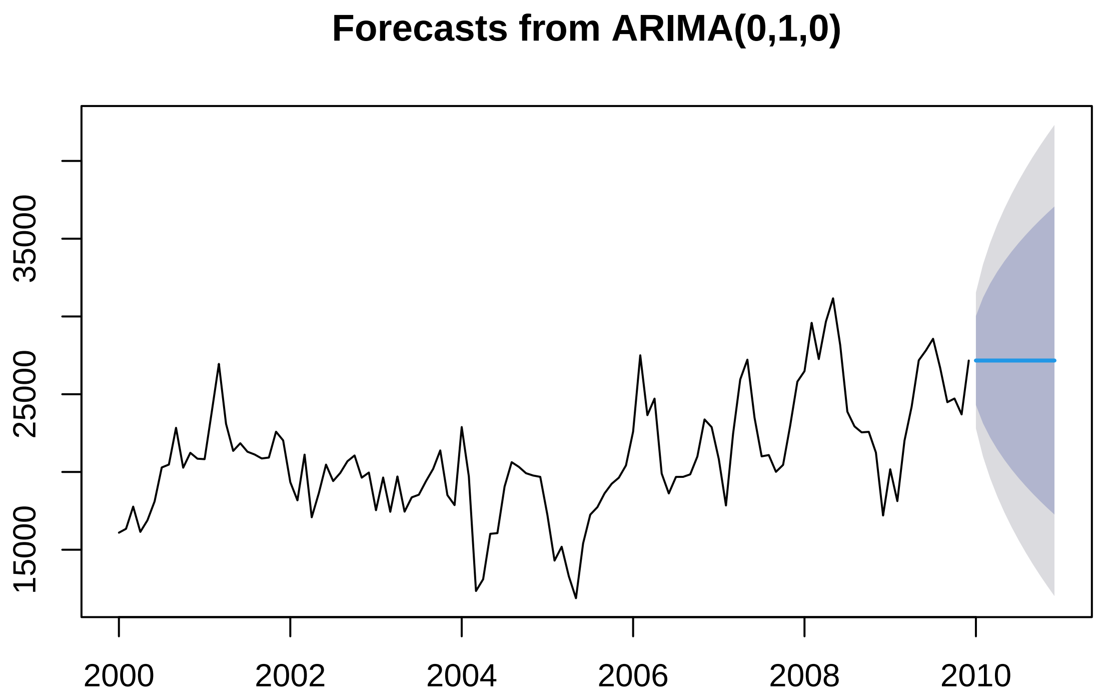
Answer: Compared with the original data, the de-seasonal data has less seasonal fluctuation in the ACF graph but still has a certain seasonality. And the deseasonalized series has better PACF due to less lag value higher than the significance line.

Part II: Forecasting with ARIMA models and its variations

Q3

Fit a non-seasonal ARIMA(p, d, q) model using the auto.arima() function to the non-seasonal data. Forecast 12 months ahead of time using the $forecast()$ function. Plot your forecasting results and further include on the plot the last year of non-seasonal data to compare with forecasted values (similar to the plot on the lesson file for M10).

```
arima_09_desea <- auto.arima(ts_09_desea, max.D = 0, max.P = 0, max.Q = 0)
arima_09_desea_forecast <- forecast(object = arima_09_desea, h = 12)
plot(arima_09_desea_forecast)
```

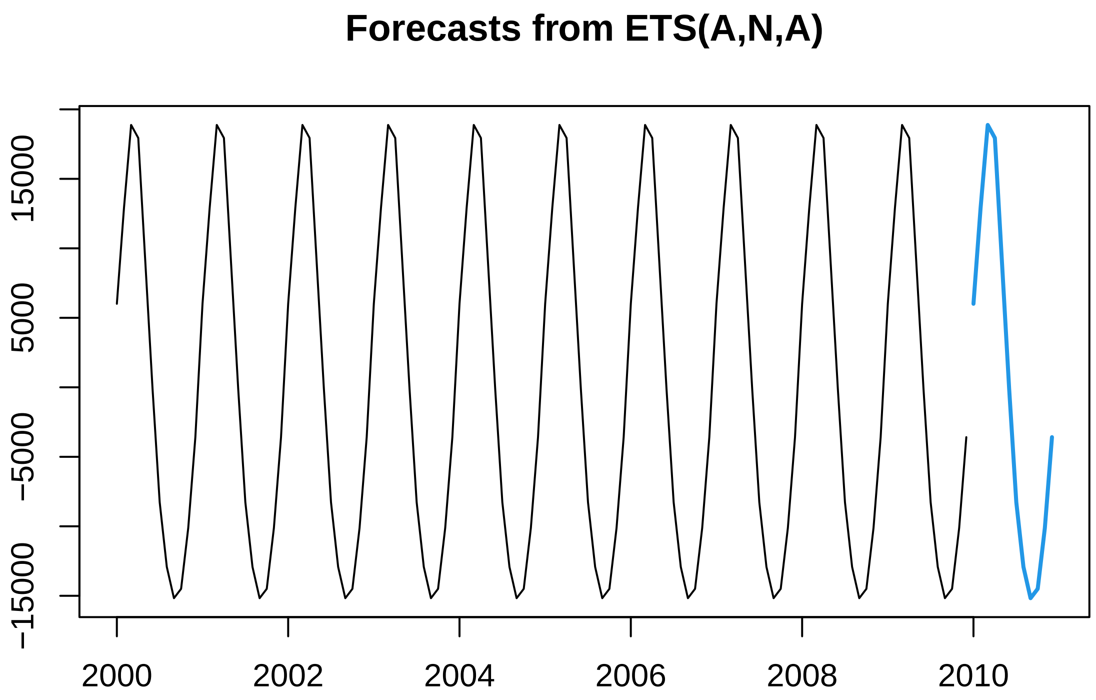


> Answer: The forecast result is a straight line, which I feel is strange and not accurate.

Q4

Put the seasonality back on your forecasted values and compare with the original seasonal data values. *Hint* : One way to do it is by summing the last year of the seasonal component from your decompose object to the forecasted series.

```
#seasonal data
inflow09_sea <- inflow09_decomp$time.series[,"seasonal"]
inflow09_sea_forecast <- forecast(inflow09_sea, h= 12)
plot(inflow09_sea_forecast)
```

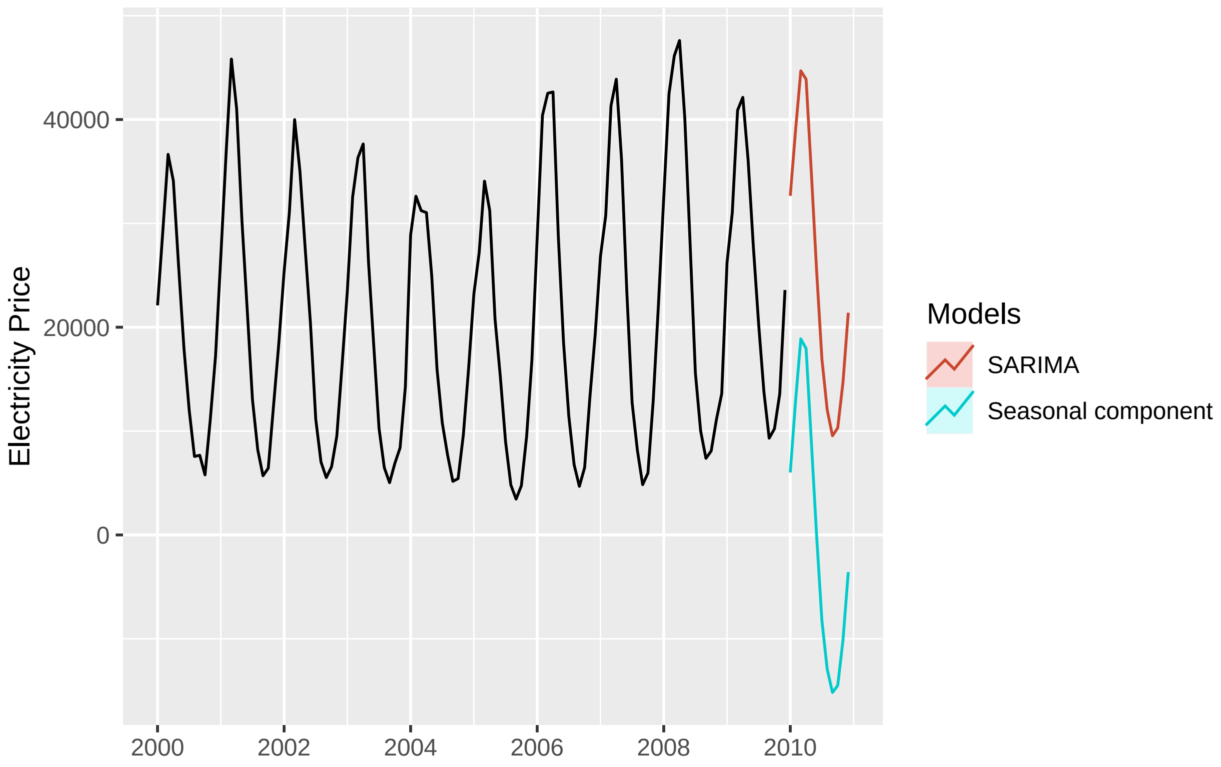


Q5

Repeat Q3 for the original data, but now fit a seasonal ARIMA$(p, d, q)x(P, D, Q)_{12}$ also using the auto.arima().

```
sarima_ts_09 <- auto.arima(ts_09)
sarima_ts_09_forecast <- forecast(object = sarima_ts_09, h = 12)

autoplot(ts_09) +
    autolayer(sarima_ts_09_forecast,series="SARIMA",PI=FALSE) +
    autolayer(inflow09_sea_forecast,series="Seasonal component",PI=FALSE) +
    ylab("Electricity Price") +
    xlab("") +
    labs(col="Models")
```



```
##note: forecast have more value, if use ggplot need identify the forecast$mean.
```

Q6

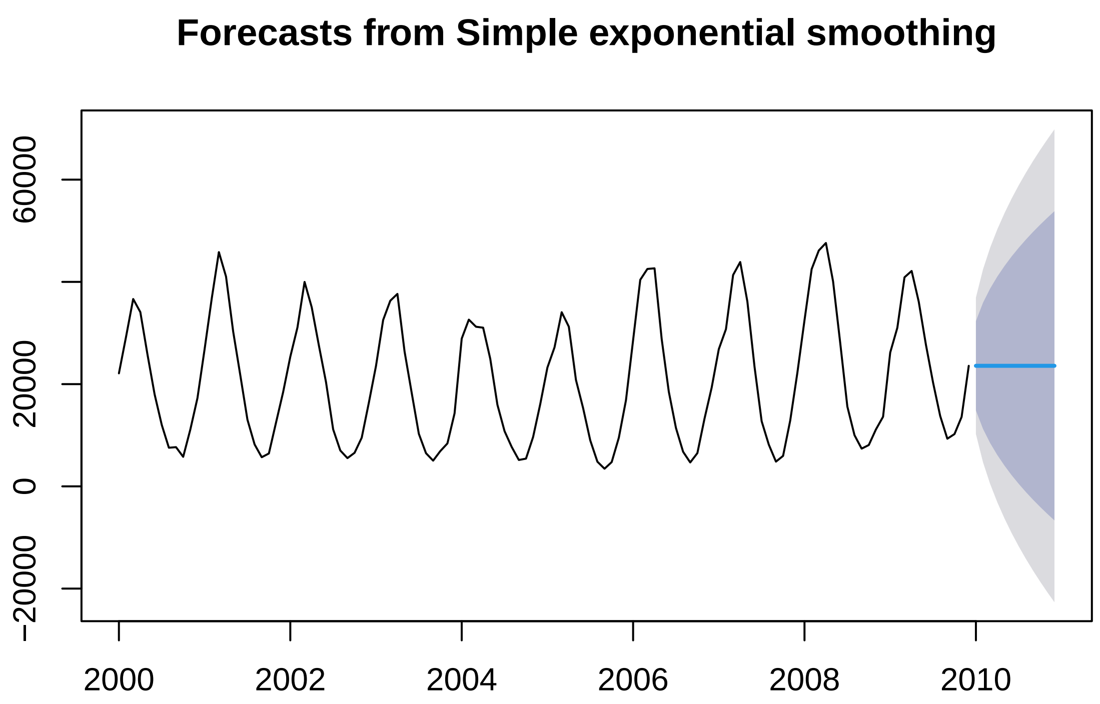
Compare the plots from Q4 and Q5 using the autoplot() function. > Answer:The fluctuations of the two forecast result are consistent. Because SARIMA includes trends and residuals, its value is closer to the real vale.

Part III: Forecasting with Other Models

Q7

Fit an exponential smooth model to the original time series using the function *ses*() from package `forecast`. Note that this function automatically do the forecast. Do not forget to set the arguments: silent=FALSE and holdout=FALSE, so that the plot is produced and the forecast is for the year of 2010.

```
ses_ts_09 <- ses(y = ts_09, h = 12, holdout = FALSE, silent = FALSE)
plot(ses_ts_09)
```



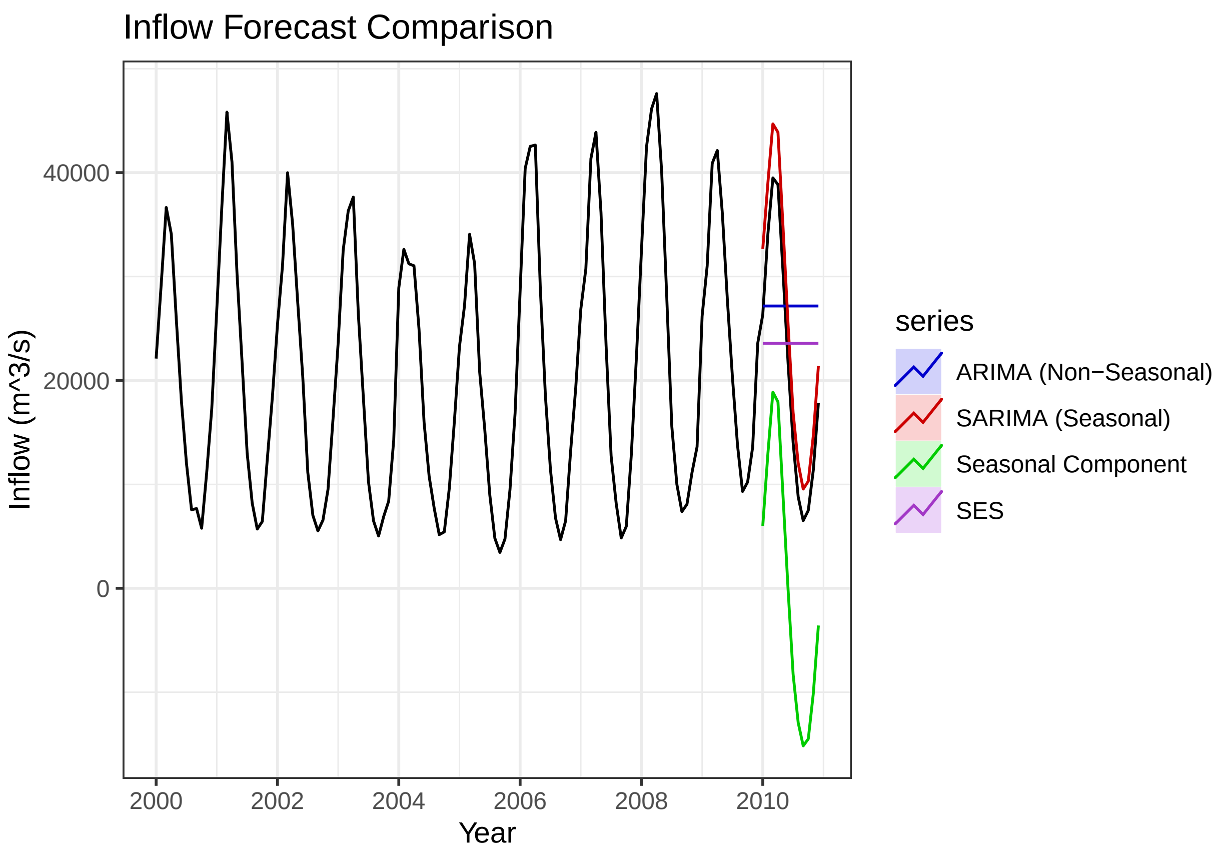
Part IV: Checking Forecast Accuracy

Q8

Make one plot with the complete original seasonal historical data (Jan 2000 to Dec 2010). Now add the forecasts from each of the developed models in parts Q4, Q5, Q7 and Q8. You can do it using the autoplot() combined with autolayer(). If everything is correct in terms of time line, the forecasted lines should appear only in the final year. If you decide to use ggplot() you will need to create a data frame with all the series will need to plot. Remember to use a different color for each model and add a legend in the end to tell which forecast lines corresponds to each model.

```r
ts_10 <- ts(raw_inflow_10$V15, start = c(2000,1), frequency = 12)


autoplot(ts_10) +
  autolayer(arima_09_desea_forecast, series = "ARIMA (Non-Seasonal)", PI = FALSE) +
  autolayer(sarima_ts_09_forecast, series = "SARIMA (Seasonal)", PI = FALSE) +
  autolayer(ses_ts_09, series = "SES", PI = FALSE) +
  autolayer(inflow09_sea_forecast, series = "Seasonal Component", PI = FALSE) +
  xlab("Year") + ylab("Inflow (m^3/s)") +
  ggtitle("Inflow Forecast Comparison") +
  scale_color_manual(values = c("blue", "red", "green", "purple")) +
  theme_bw()
```



Q9

From the plot in Q9 which model or model(s) are leading to the better forecasts? Explain your answer. Hint: Think about which models are doing a better job forecasting the high and low inflow months for example.

> Answer:SARIMA model is best. It closely follows the original inflow and captures the seasonal fluctuations.

Q10

Now compute the following forecast metrics we learned in class: RMSE and MAPE, for all the models you plotted in part Q9. You can do this by hand since your have forecasted and observed values for the year of 2010. Or you can use R function *accuracy*() from package "forecast" to do it. Build and a table with the results and highlight the model with the lowest MAPE. Does the lowest MAPE corresponds match your answer for part Q10?

```
observed_2010 <- raw_inflow_2010[,"V15"]
accuracy_arima <- accuracy(arima_09_desea_forecast$mean, observed_2010)
accuracy_sarima <- accuracy(sarima_ts_09_forecast$mean, observed_2010)
accuracy_ses <- accuracy(ses_ts_09$mean, observed_2010)
accuracy_sea <- accuracy(inflow09_sea_forecast$mean, observed_2010)

scores <- as.data.frame(rbind(accuracy_arima, accuracy_sarima, accuracy_ses,accuracy_sea ))
row.names(scores) <- c("ARIMA", "SARIMA", "SES","SeasonalC")
```

Table 1: Forecast Accuracy

	ME	RMSE	MAE	MPE	MAPE
ARIMA	-5750.384	13027.497	11454.128	-84.02912	99.43078
SARIMA	-4031.818	4171.259	4031.818	-23.87315	23.87315
SES	-2165.000	11888.481	10718.167	-59.74209	83.65282
SeasonalC	21417.000	21424.024	21417.000	146.44650	146.44650

```
kbl(scores,
      caption = "Forecast Accuracy",
      digits = array(5,ncol(scores)))
```

```
best_model_index <- which.min(scores[,"MAPE"])
cat("The best model by MAPE is:", row.names(scores[best_model_index,]))
```

```
## The best model by MAPE is: SARIMA
```

Answer:Yes, it is SARIMA.

##window function-